Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070532

Security and Improvement of an Authenticated Group Key Transfer Protocol Based on Secret Sharing

Wei Yuan[1,*], *Liang Hu*[2], *Hongtu Li*[2] *and Jianfeng Chu*[2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Science, Beijing, China
[2] School of Computer Science and Technology, Jilin University, Changchun 130012, China

Received: 25 Jan. 2013, Revised: 26 May. 2013, Accepted: 27 May. 2013
Published online: 1 Sep. 2013

Abstract: Recently Harn and Lin proposed a novel authenticated group key transfer protocol that a mutually trusted key generation center (KGC) can broadcast group key information to all group members at once and only authorized group members can recover the group key. This paper presents that Harn and Lin's protocol can not withstand man-in-the-middle attack and describes the reasons and detailed processes that the group key is gained by the active attacker who is not included in the member list of that particular group. To fill the gaps, we discuss the problems, possible solutions, and propose an improved protocol.

Keywords: Group key transfer, secret sharing, authentication, cryptanalysis

1 Introduction

Key transfer protocols [1–3] and key agreement protocols [4–6] are two types of key establishment protocols. Key transfer protocols rely on a mutually trusted key generation center (KGC) to select session keys and then transport session keys to all communication entities secretly. Most often, KGC encrypts session keys under another secret key shared with each entity during registration. In key agreement protocols [7, 8], all communication entities are involved to determine session keys. The most commonly used key agreement protocol is Diffie-Hellman (DH) key agreement protocol [9]. In DH protocol, the session key is determined by exchanging public keys of two communication entities. Most key transfer protocols [10–16] take natural generalization of the DH key agreement protocol. There are other key transfer protocols based on non-DH key agreement approach as well. Tzeng [17] proposed a conference key agreement protocol based on discrete logarithm (DL) assumption with fault tolerance in recent years. In 2008, Cheng and Laih [18] modified Tseng's conference key agreement protocol based on bilinear pairing. In 2009, Huang [19] proposed a noninteractive protocol based on DL assumption to improve the efficiency of Tseng's protocol.

In 1989, Laih et al. [20] proposed the first algorithm using any (t,n) secret sharing scheme to distribute a group key to a group consisting of t-1 members. It is obvious that the scheme using this approach is more efficient than encrypting and distributing the group key to each member of the group. Later, there are some papers [21–23] following the same concept to distribute group messages to multiple users. The newest research owes Lein Harn and Changlu Lin. They summarize the approaches in these papers and proposed a novel protocol [1] based on secret sharing scheme [24] for distributing group key. Their protocol is rather simple and efficient. However, the initial conditions in their protocol are not very strict. If the protocol execute normally, the security of their transformation is really information theoretically secure. Due to lack of the authentication at the beginning, the latter secure verification can be bypassed.

In this paper, we show that the attacker, who is not included in the list of a particular group, can impersonate any group member to join in that group only if the attacker outside of that group is allowed to request for group key service in their protocol. This condition is a basic feature to everyone who wants to make use of their protocol. To solve this problem, we add some additional verification operations. The analysis shows the users who

* Corresponding author e-mail: yuanwei1@126.com

have subscribed the key distribution service but not are included in a particular group can not join in that group furtively. The rest of this paper is organized as follows. Section 2 briefly reviews Harn and Lin's key transfer protocol. Section 3 proposes an attack to their protocol. Section 4 discusses the problems and presents an improved authenticated group key transfer protocol. Section 5 demonstrates the security of our improved protocol. A conclusion is made in section 6.

2 Brief introduction of the original authenticated group key transfer protocol

Harn et al.′s authenticated group key transfer protocol consists of three processes: initialization of KGC, user registration, and group key generation and distribution. Fig.1 describe the protocol. The detail steps are as follows:

Initialization of KGC. The KGC randomly chooses two primes p and q and computes $n = p \times q$. n is published.

User Registration. Each user is required to register at the KGC for subscribing the key distribution service. The KGC keeps tracking all the registered users and removing any unsubscribed users. During registration, KGC shares a secret, (x_i, y_i), with each user, U_i, where $x_i, y_i \in Z_n^*$.

Group key generation and distribution. Upon receiving group key generation request from any user, KGC needs to randomly select a group key and access all the shared secrets with the group members. KGC needs to distribute this group key to all the group members in a secure and authenticated way. All the communications between KGC and group members are in a broadcast channel. For example, we assume that a group consists of t members, $\{U_1, U_2, \cdots, U_t\}$, and shared secrets are (x_i, y_i), for $i = 1, \cdots, t$. The key generation and distribution process contains five steps.

Step1. The initiator sends a key generation request to KGC with a list of group members as $\{U_1, U_2, \cdots, U_t\}$.

Step2. KGC broadcasts the list of all the participating members, $\{U_1, U_2, \cdots, U_t\}$, as a response.

Step3. Each participating group member needs to send a random challenge, $R_i \in Z_n^*$, to KGC.

Step4. KGC randomly selects a group key, k, and generates an interpolated polynomial $f(x)$ with degree t to pass through $(t+1)$ points, $(0, k)$ and $(x_i, y_i \oplus R_i)$, for $i = 1, \cdots, t$. KGC also computes t additional points, $P_1, \cdots, P_t$, on f(x) and $Auth = h(k, U_1, U_2, \cdots, U_t, R_1, R_2, \cdots, R_t, P_1, P_2, \cdots, P_t)$, where h is a one-way hash function. All the computations on $f(x)$ are over Z_n^*. KGC broadcasts $\{Auth, P_1, \cdots, P_t\}$ to all the group members. All the computations are performed in Z_n^*.

Step5. For each group member U_i, knowing the shared secret, $(x_i, y_i \oplus R_i)$, and t additional public points, P_i, for $i = 1, \cdots, t$, on $f(x)$, he is able to compute the polynomial $f(x)$ and recover the group key $k = f(0)$. Then, U_i computes $h(k, U_1, U_2, \cdots, U_t, R_1, R_2, \cdots, R_t, P_1, P_2, \cdots, P_t)$ and checks whether this hash value is identical to $Auth$. If these two values are identical, U_i authenticates the group key sent from KGC.

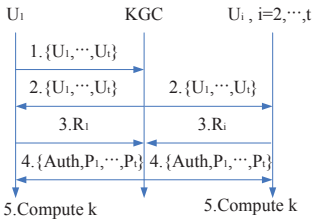

Fig. 1: Simple description of Harn et al.′s protocol

3 Proposed attack

In above protocol that we will study, simultaneous broadcasts are intensively used. However it is actually a multi-cast, in which the attacker may delay, modify, or cancel the message sent to each recipient independently [25].

Suppose an attacker want to make an active attack to impersonate a group member. Her aim is to obtain the group key and attend their secret conference. She has the ability to intercept messages between the KGC and normal group members and can forge a new one as well. To abide by the protocol, she should get the published parameter n and subscribe the key distribution service of the KGC before the attack. Suppose her general identity is U_e and the shared secret between the KGC and her is (x_e, y_e), where $x_e, y_e \in Z_n^*$. Note that her general identity is not included in the list of the group members, who want to start a conversation. That is $e \notin [1, t]$.Attack processes are described as follows:

1. U_e intercepts the key generation request, which contains a list of group members as $\{U_1, U_2, \cdots, U_t\}$ to the KGC. Then U_e deletes any one, such as U_i where $i \in [1, t]$, in the list $\{U_1, U_2, \cdots, U_t\}$, and replaces U_i with her identity U_e in the forged list. Finally, she unicasts the forged list $\{U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t\}$ to the KGC.

2. U_e intercepts the response, $\{U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t\}$, from the KGC, and broadcasts the original list $\{U_1, U_2, \cdots, U_t\}$ to all the participating members.

After the two steps above, the KGC believes the participating members are $\{U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t\}$, but group members consider $\{U_1, \cdots, U_{i-1}, U_i, U_{i+1}, \cdots, U_t\}$ are going to start a new conversation.

3. U_e intercepts $R_i \in Z_n^*$ from U_i and unicasts her random challenge $R_e \in Z_n^*$ to the KGC. At the same time, U_e records all the R_js to the KGC, where $j = 1, \cdots, t, j \neq i$.

In the step4 of original protocol, KGC will compute $f(x)$ with the $t+1$ points $(x_j, y_j \oplus R_j)$, where $j = 1, \cdots, t$, $j \neq i$, $(x_e, y_e \oplus R_e)$ and $(0,k)$. Then KGC computes t additional points $P_1, \cdots, P_t$ on $f(x)$, computes $Auth = h(k, U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t, R_1, \cdots, R_{i-1}, R_e, R_{i+1}, \cdots, R_t, P_1, P_2, \cdots, P_t)$ and broadcasts $\{Auth, P_1, \cdots, P_t\}$.

4. U_e intercepts $\{Auth, P_1, \cdots, P_t\}$ sent from the KGC, where $Auth = h(k, U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t, R_1, \cdots, R_{i-1}, R_e, R_{i+1}, \cdots, R_t, P_1, P_2, \cdots, P_t)$ Then she computes $(x_e, y_e \oplus R_e)$ with her challenge R_e and her own secret value (x_e, y_e). The group key k can be computed with the $t+1$ points $(P_1, \cdots, P_t)$ and $(x_e, y_e \oplus R_e)$. Finally, she forges the signature $Auth' = h(k, U_1, \cdots, U_{i-1}, U_i, U_{i+1}, \cdots, U_t, R_1, \cdots, R_{i-1}, R_i, R_{i+1}, \cdots, R_t, P_1, P_2, \cdots, P_t)$, and broadcasts $\{Auth', P_1, \cdots, P_t\}$ to all the group members except U_i. In the step 5 of the original protocol, each group member U_j, where $j = 1, \cdots, t, j \neq i$, is able to compute the group key k with $(P_1, \cdots, P_t)$ and $(x_j, y_j \oplus R_j)$. Then U_j computes the hash value $h(k, U_1, \cdots, U_{i-1}, U_i, U_{i+1}, \cdots, U_t, R_1, \cdots, R_{i-1}, R_i, R_{i+1}, \cdots, R_t, P_1, P_2, \cdots, P_t)$ with the member list that he reserves in the step1 and compares it with the received $Auth'$. Since these two values are identical, U_j accepts the group key k. As a result, $U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t$ will start a new conversation and U_i can not obtain the group key.

In the end of the Harn et al.′s protocol, they claim their protocol does not focus on user authentication and messages authentication that from group members to KGC. But they suggest that the following two additional steps can achieve above two features.

First, in step3 of the original protocol, each user U_i attaches an authentication value, $h((x_i, y_i), R_i)$, along with the challenge message R_i. Then KGC can authenticate R_i. Second, after step5 of the original protocol, each user U_i sends a key confirmation, $h((x_i, y_i), k)$, to KGC. Then, after receiving all key confirmations, KGC sends a group key confirmation, $h((x_i, y_i), k, U_1, \cdots, U_t)$, to each group member. As the result, each user U_i can confirm the group key. The protocol with key confirmation can be described as Fig.2.

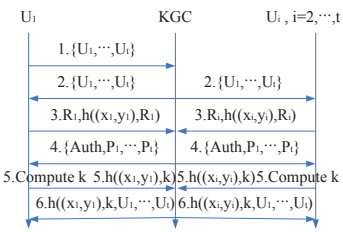

Fig. 2: Harn et al.′s protocol with key confirmation

It seems that the protocol with the additional key confirmation steps can prevent our man-in-the-middle attack, because the key confirmation $h((x_i, y_i), k, U_1, \cdots, U_t)$ contains the user list $\{U_1, \cdots, U_t\}$ and the shared secret between each user and the KGC. The attacker U_e can not forge valid key confirmation $h((x_i, y_i), k, U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t)$ without the shared secret (x_i, y_i). However, actually, these steps not only do not enhance the security of the original protocol, but also lead their protocol suffers from more serious attacks. Suppose KGC sends a group key confirmation $h((x_1, y_1), k, U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t)$ to a user U_1 after step5. The attacker U_e intercepts it and does not forward it to U_1 immediately. Since the group key k has been computed and the user list $\{U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t\}$ is known to her, she can guess a pair of number (x_1', y_1') and verify whether it is U_1's secret by the equation $H \stackrel{?}{=} h(x_1', y_1', k, U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t)$ in an offline manner. $H = h((x_1, y_1), k, U_1, \cdots, U_{i-1}, U_e, U_{i+1}, \cdots, U_t)$ is the intercepted key confirmation. As the result, U_e will get U_1's secret and thus she can impersonate U_1 directly. It means that adding these additional steps may lead the user′s secret reveals.

Actually, these two additional steps are just a suggestion at the last of the original paper. The security theorems in the original paper even do not analyze their validity. Hence, we do not consider them in our attack.

After four steps attack, the outside attacker U_e can impersonate U_i to participate in the new conversation with other group members and U_i will be kicked out off the group. Since U_i may be any one of the group member, U_e can impersonate any one she wants to replace.

However, if the attacker is not familiar with others, she may not have enough knowledge to talk with each others. Even if she owns the group key, other members may find she is not U_i by the content in the conversation.

To overcome this shortage, the attacker can continue the attacking process as follows:

5. U_e unicasts a new key generation request to the KGC with the group members $\{U_e, U_2, \cdots, U_i, \cdots, U_t\}$.

6. U_e intercepts the response $\{U_e, U_2, \cdots, U_i, \cdots, U_t\}$ from the KGC. For the response has been sent to U_i, U_e does not need to unicast another list.

7. U_e unicasts the challenge $R_e \in Z_n^*$ and R_j, where $j = 2, \cdots, t$, to the KGC. Note, R_j is the original challenge intercepted from U_j in the step3.

In the step4 of original protocol, KGC will compute $f(x)$ with the $t+1$ points $(x_j, y_j \oplus R_j)$, where $j = 2, \cdots, t$, $(x_e, y_e \oplus R_e)$ and $(0, k_e)$. Then KGC will compute t additional points $P_1, \cdots, P_t$ on $f(x)$, computes $Auth = h(k, U_e, U_2, \cdots, U_i, \cdots, U_t, R_e, R_2, \cdots, R_i, \cdots, R_t, P_1, P_2, \cdots, P_t)$ and broadcasts $\{Auth, P_1, \cdots, P_t\}$.

8. U_e intercepts $\{Auth, P_1, \cdots, P_t\}$ sent from the KGC, where $Auth = h(k, U_e, U_2, \cdots, U_i, \cdots, U_t, R_e, R_2, \cdots, R_i, \cdots, R_t, P_1, P_2, \cdots, P_t)$. Then she computes the group key

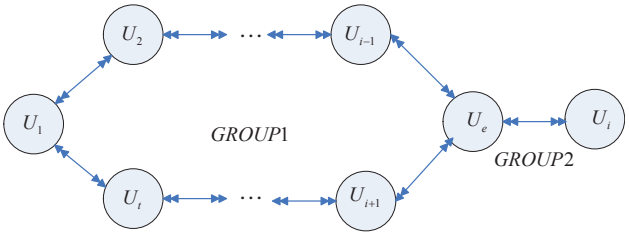

Fig. 3: Result after step8

k_e with the t+1 points $(P_1,\cdots,P_t)$ and $(x_e,y_e \oplus R_e)$. Finally, she generates a new signature $Auth' = h(k_e,U_1,\cdots,U_i,\cdots,U_t,R_1,\cdots,R_i,\cdots,R_t,P_1,P_2,\cdots,P_t)$ and unicasts $\{Auth',P_1,\cdots,P_t\}$ to U_i. After receiving the signature $Auth'$ and the points P_i, U_i can compute k_e with $(P_1,\cdots,P_t)$ and $(x_j,y_j \oplus R_j)$. Then he computes the hash value $h(k_e,U_1,\cdots,U_i,\cdots,U_t,R_1,\cdots,R_i,\cdots,R_t,P_1,P_2,\cdots,P_t)$ with the member list that he reserves in the step1 and compares it with the received $Auth'$. Since these two values are identical, U_i accepts the group key k_e. As a result, $\{U_i,U_e\}$ will start a new conversation and U_e can gain enough knowledge to talk with other t-1 participators. As the result of all 8 steps, U_e participates in two conversations at the same time. One is with $U_1,\cdots,U_{i-1},U_{i+1},\cdots,U_t$, another is with U_i. The result can be described as Fig. 3.

When someone in the group1 talks something that the attacker does not know, she can send this message to U_i and give back U_i's response as her response. In addition, if she believes U_i's response doesn't meet her needs, she can also forge another one based on U_i's response. For example: A company has 10 departments, each department owns 9 employees and 1 supervisor. If the company uses this protocol to distribute group keys, all 90 employees and 10 supervisors should subscribe the key distribution service and each department can form a regular group and use the group key to deal with their own vocational work confidentially. However, when the supervisors want to form a group and talk some secrets, any employee can eavesdrop or tamper on it with above method. Finally, a conclusion can be made that anyone who has established a shared secret with KGC can obtain the group key; she does not need to be a member of that group. An example to attack three members' group is described as Fig. 4.

4 Problem discussion and improvement

Due to lack of user authentication in the step1 and the step2, attackers can modify the group member list and both KGC and group members can not verify it. It is the main reason leading to our proposed attack. In the last of the original paper, the authors discussed user authentication and authenticated message transmitted from group members to the KGC. However, their attentions mainly focus on the 3^{rd} step and the 5^{th} step. If the user has been kicked out of the group in the step1 and the step2, all the later efforts are useless.

It seems that the share secret value (x, y) may help achieve user authentication, which is discussed in the original paper. That is, if the initiator attaches $h(U_1,U_2,\cdots,U_t,(x_1,y_1))$ to the member list $\{U_1,U_2,\cdots,U_t\}$, which the authors mentioned in the remark 2 of the original paper. Unfortunately, this improvement can not withstand proposed attack, because the attacker can kick the initiator out by replacing $\{U_1,U_2,\cdots,U_t,h(U_1,U_2,\cdots,U_t,(x_1,y_1))\}$ with $\{U_e,U_2,\cdots,U_t,h(U_e,U_2,\cdots,U_t,(x_e,y_e))\}$.

One possible way to solve this problem is to narrow the scope of the protocol. That is, only authenticated users are allowed to subscribe the key distribution service in user registration process and all users registered at the KGC form only one group. When the conversation ends, all users must unsubscribe the key distribution service. Then the request list and response list in step1 and step2 can be omitted and our proposed attack is sure to fail, because there is only one group and KGC can sure all members' identity in the group. However, how to authenticate a user in the user registration process becomes a new problem and the author's goal is to supply user authentication in this protocol. Even if we can construct a new authentication protocol to forbid unauthenticated users to subscribe the key distribution service, security risk still exist. (e.g. if the group membership changes, a person who is no longer a member of a designated group can also have access to the group key as long as his shared secret key with KGC is still valid).

Another way to solve this problem is to add effective verification functions in step1 or step2. So that any change in the request list or response list can be found by group members or the KGC. Our suggested protocol will follow this idea.

Suppose a user U_{i+1} has registered at the KGC and wants to agree on a group key with $\{U_{t+2},U_{t+3},\cdots,U_{2t}\}$. Then U_{i+1} is not an outsider but it is also not a member of the particular group $\{U_1,U_2,\cdots,U_t\}$, hence the theorem 1 of the original paper does not prove the key confidentiality for lacking of a kind of users as U_{i+1}. So our improvement will mainly focus on these *intermediate users* as U_{i+1}. During the execution of our improvement protocol, the attacker has the entire control of the network, and tries her best to break the privacy of the key.

The proposed improvement protocol consists of three processes: initialization of the KGC, user registration, and group key generation and distribution. The detailed description is as follows:

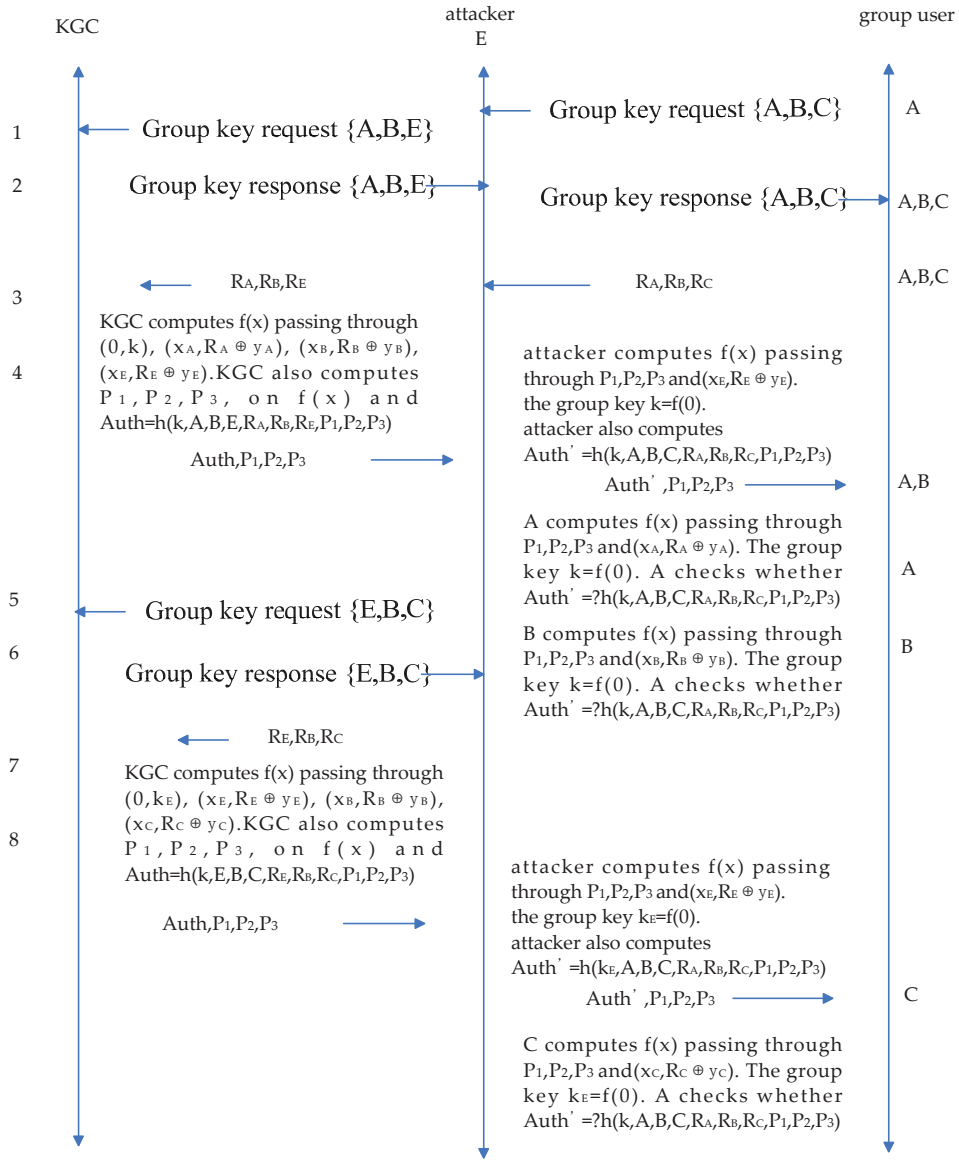

Fig. 4: Attack process to a three member's group

Initialization of KGC. The KGC randomly chooses two safe primes p and q, and computes $n = p \times q$. Then the KGC selects a number $e \in Z_n^*$, which demands e and (p-1)×(q-1) are coprime. In additions, the KGC computes another number d, which meets $e \times d \equiv 1$ mod (p-1)×(q-1).

User Registration. Each user is required to register at the KGC for subscribing the key distribution service. The KGC keeps tracking all the registered users and removing any unsubscribed users. During the registration, the KGC shares (e,n) and x_i, y_i with each user U_i, where $x_i, y_i \in Z_n^*$.

Group key generation and distribution. Upon receiving group key generation request from any user, KGC needs to randomly select a group key and access all the shared secrets with the group members. KGC needs to distribute this group key to all the group members in a secure and authenticated way. All the communications between KGC and group members are in a broadcast channel. For example, we assume that a group consists of t members, $\{U_1, U_2, \cdots, U_t\}$, and shared secrets are (x_i, y_i), for $i = 1, \cdots, t$. The key generation and distribution process contains five steps.

Step1. The initiator sends a key generation request to KGC with a list of group members as $\{U_1, U_2, \cdots, U_t\}$.

Step2. KGC broadcasts the list of all the participating members and a value $\{U_1, U_2, \cdots, U_t, v\}$ as a response, where $v = h(U_1, U_2, \cdots, U_t)^d \ mod \ n$.

Step3. After receiving $\{U_1, U_2, \cdots, U_t, v\}$, each participating group member should compute $h(U_1, U_2, \cdots, U_t)$ and checks whether this value is identical to $v^e \ mod \ n$. If these two values are identical, he needs to send a random challenge, $R_i \in Z_n^*$, to the KGC.

Step4. The KGC randomly selects a group key k and generates an interpolated polynomial $f(x)$ with degree t to pass through $(t+1)$ points, $(0,k)$ and $(x_i, y_i \oplus R_i)$, for $i = 1, \cdots, t$. The KGC also computes t additional points, P_i, for $i = 1, \cdots, t$, on $f(x)$ and $Auth = h(k, U_1, \cdots, U_t, R_1, \cdots, R_t, P_1, \cdots, P_t)$, where h is a one-way hash function. All the computations on $f(x)$ are over Z_n^*. The KGC broadcasts $\{Auth, P_1, \cdots, P_t\}$ to all the group members. All the computations are performed in Z_n^*.

Step5. Each group member U_i, knowing the shared secret, $(x_i, y_i \oplus R_i)$, and t additional public points, P_i, for $i = 1, \cdots, t$, on $f(x)$, is able to compute the polynomial $f(x)$ and recovers the group key $k = f(0)$. Then U_i computes $h(k, U_1, \cdots, U_t, R_1, \cdots, R_t, P_1, \cdots, P_t)$ and checks whether this hash value is identical to $Auth$. If these two values are identical, U_i authenticates the group key sent from the KGC.

5 Security analysis

Comparing with the original protocol, our protocol only modifies the step2 and step3 of the group key generation and distribution process. The verifications to against the outside attacks and insider attacks are kept in our improvement. Our modifications do not weaken the security in key confidentiality, and key authentication. Thus, our improvement can be regarded as a consummation of the theorem 1 in the original protocol.

Theorem 1 In our protocol, the *intermediate users* can not replace any group user in the group list without being detected.

Proof. In our protocol, the KGC adds a public key e and a private key d in the initialization process and sends them to all the registered users during registration process. As a result, each user knows the public key of the KGC, no matter who participates in a certain group distribution process. If an intermediate user wants to initiate an attack described above, she should intercept the key generation request, which contains a list of group members as $\{U_1, U_2, \cdots, U_t\}$ to the KGC and replace U_i with her identity U_e in the forged list. However, she can not generate a valid collision to the response list $U_1, \cdots, U_{i-1}, U_e,$
$U_{i+1}, \cdots, U_t, v$, because d is the private key of the KGC and $v = h(U_1, U_2, \cdots, U_t)^d \ mod \ n$. In the step3 of our protocol, each group member can detect that the member list has been modified.

6 Conclusions

Due to the attack described above, Harn et al.′s authenticated group key transfer protocol based on secret sharing doesn't achieve their goals. We discuss that problem and propose an improved protocol. Security analysis shows that our improved protocol has modified the flaws and any one outside of a particular group can not gain the group key without being detected.

References

[1] Lein Harn and Changlu Lin, Authenticated Group Key Transfer Protocol Based on Secret Sharing, IEEE Trans. Computers, **59**, 842 (2010).

[2] N. W. Lo, Kuo-Hui Yeh, Cryptanalysis of two three-party encrypted key exchange protocols, Computer Standards & Interfaces, **31**, 1167 (2009).

[3] V. A. Ustimenko, Y. M. Khmelevsky, Walks on graphs as symmetric or asymmetric tools to encrypt data, The South Pacific Journal of Natural and Applied Sciences, **20**, 34-44 (2002).

[4] Xianfeng Guo, Jiashu Zhang, Secure group key agreement protocol based on chaotic Hash, Information Sciences, **180**, 4069 (2010).

[5] Wei Yuan, Liang Hu, Hongtu Li, Jianfeng Chu, An Efficient Password-based Group Key Exchange Protocol Using Secret Sharing, applied mathematics & information sciences, **7**,145 (2013).

[6] Ming-Hui Zheng, Hui-Hua Zhou, Jun Li, Guo-Hua Cui, Efficient and provably secure password-based group key agreement protocol, Computer Standards & Interfaces, **31**, 948 (2009).

[7] Jung Yeon Hwang, Kyu Young Choi, Dong Hoon Lee, Security weakness in an authenticated group key agreement protocol in two rounds, Computer Communications, **31**, 3719 (2008).

[8] Yuh-Min Tseng, A resource-constrained group key agreement protocol for imbalanced wireless networks, Computers and Security, **26**, 331 (2007).

[9] W. Diffie and M. E. Hellman, New Directions in Cryptography, IEEE Trans. Information Theory, **22**, 644 (1976).

[10] M. Burmester and Y. G. Desmedt, A Secure and Efficient Conference Key Distribution System, Proc. Eurocrypt '94 Workshop Advances in Cryptology, 275-286 (1994).

[11] Sandeep S. Kulkarni, Bezawada Bruhadeshwar, "Key-update distribution in secure group communication," Computer Communications, **33**, 689 (2010).

[12] I. Ingemarsson, D. T. Tang, and C. K. Wong, A Conference Key Distribution System, IEEE Trans. Information Theory, **28**, 714, (1982).

[13] Lu RX, Cao ZF. Simple three-party key exchange protocol, Computers and Security, **26**, 94 (2007).

[14] D. G. Steer, L. Strawczynski, W. Diffie, and M. J. Wiener, A Secure Audio Teleconference System, Proc. Eighth Ann. Int'l Cryptology Conf. Advances in Cryptology (Crypto), **88** 520-528 (1988).

[15] M. Steiner, G. Tsudik, and M. Waidner, Diffie-Hellman Key Distribution Extended to Group Communication, Proc. Third ACM Conf. Computer and Comm. Security (CCS), **96**, 31-37 (1996).
[16] Weichao Wang, Tylor Stransky, Stateless key distribution for secure intra and inter-group multicast in mobile wireless network, Computer Networks, **51**, 4303 (2007).
[17] W. G. Tzeng, A Secure Fault-Tolerant Conference Key Agreement Protocol, IEEE Trans. Computers, **51**, 373 (2002).
[18] J. C. Cheng and C. S. Laih, Conference Key Agreement Protocol with Non Interactive Fault-Tolerance Over Broadcast Network, Int. J. Information Security, **8**, 37 (2009).
[19] Jianjie Zhao, Dawu Gu, Yali Li, An efficient fault-tolerant group key agreement protocol, Computer Communications, **33**, 890 (2010).
[20] C. Laih, J. Lee, and L. Harn, A New Threshold Scheme and Its Application in Designing the Conference Key Distribution Cryptosystem, Information Processing Letters, **32**, 95 (1989).
[21] S. Berkovits, How to Broadcast a Secret, Proc. Eurocrypt '91 Workshop Advances in Cryptology, 536-541 (1991).
[22] C. H. Li and J. Pieprzyk, Conference Key Agreement from Secret Sharing, Proc. Fourth Australasian Conf. Information Security and Privacy (ACISP '), **99**, 64 (1999).
[23] G. Saze, Generation of Key Predistribution Schemes Using Secret Sharing Schemes, Discrete Applied Math, **128**, 239 (2003).
[24] A. Shamir, "How to Share a Secret," Comm. ACM, **22**, 612 (1979).
[25] Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, and Pointcheval, Password-based Group key Exchange in a Constant Number of Rounds, PKC2006, LNCS, **3958**, 427-442.

Wei Yuan was born in Chengde of Hebei province of China in 1984. He began the study of computer science at Jilin University in 2003 and got his bachelor degree in 2007. Then he continued his research on information security and received his master degree in 2010. Now he is a Ph.D. candidate of the college of computer science and technology of Jilin University. His main research interests include cryptography and information security. he have participated in several projects include two National Natural Science Foundations of China and one National Grand Fundamental Research 973 Program of China and published more than 20 research papers from 2007.

Liang Hu was born in 1968. He has his BS degree on Computer Systems Harbin Institute of Technology in 1993 and his Ph.D. on Computer Software and Theory in 1999. Currently, he is the professor and Ph.D. supervisor of College of Computer Science and Technology, Jilin University, China. His main research interests include distributed systems, computer networks, communications technology and information security system, etc. As a person in charge or a principal participant, Dr Liang Hu has finished more than 20 national, provincial and ministerial level research projects of China.

Li Hongtu was born in Siping of Jilin, China on Mar. 17 1984. In 2002, Li Hongtu began the study of computer science at Jilin University in Jilin, Changchun, China. And in 2006, Li Hongtu got bachelor's degree of computer science. In the same year, Li Hongtu began the master's degree study in network security at Jilin University. After 3 years study, Li Hongtu got his master's degree in 2009. From then on, Li Hongtu began the doctor's degree in the same field of study at the same University. From 2009, he has got a fellowship job. He worked in grid and network security laboratory as an ASSISTANT RESEACHER at Jilin University. From 2006 to now, he has published several papers.

Jianfeng Chu, born in 1978, Ph.D. , Now he is the teacher of the College of Computer Science and Technology, Jilin University, Changchun, China. He received the Ph.D. degree in computer structure from Jilin University in 2009. His current research interests focus on information security and cryptology. An important objective of the projects is to probe the trend of network security, which can satisfy the need of constructing high-speed, large-scale and multi-services networks. Various complex attacks can not be dealt with by simple defense. And to add mechanisms to network architecture results in decreasing performance. In a word, fundamental re-examination of how to build trustworthy distributed network should be made.